МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

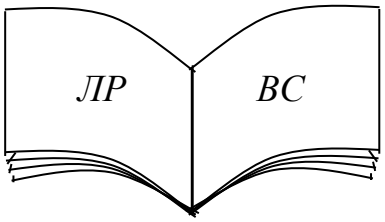
МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з навчальної дисципліни «Вбудовані системи»

для студентів денної та заочної форм навчання

за спеціальністю «Комп’ютерна інженерія»



Харків 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з навчальної дисципліни «Вбудовані системи»

для студентів денної та заочної форм навчання
за спеціальністю «Комп'ютерна інженерія»

Затверджено
редакційно-видавничою
радою НТУ «ХПІ»,
протокол № 1 від 16.01.2019 р.

Харків
НТУ «ХПІ»
2020

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Вбудовані системи» для студентів денної та заочної форм навчання за спеціальністю «Комп'ютерна інженерія» / уклад.: М. В. Ліпчанський, В. В. Скородєлов, Г. В. Гейко – Харків : НТУ «ХПІ». – 2020. – 29 с.

Укладачі: М. В. Ліпчанський, В. В. Скородєлов, Г. В. Гейко

Рецензент: доц. О. Ф. Даниленко

Кафедра обчислювальної техніки та програмування

ВСТУП

Методичні вказівки містять методику виконання лабораторних робіт, метою яких є: ознайомлення з принципами роботи мікроконтролерів та відпрацьовування практичних навичок для їх інженерного проєктування і налагодження. Методика розрахована як для застосування універсальних стендів-конструкторів із набором дискретних елементів, мікросхем та змінних модулів, так і для використання пакетів комп'ютерного моделювання.

При проведенні лабораторних робіт необхідно керуватися такими положеннями:

1) лабораторні заняття проводяться фронтально у всій групі, об'єм завдань визначає викладач;

2) до кожної лабораторної роботи необхідна самостійна підготовка, що включає вивчення теоретичного матеріалу та виконання необхідних проєктних робіт;

3) під час виконання лабораторної роботи студенти повинні виконати індивідуальне завдання та надати викладачеві всі розроблені програми та результати досліджень;

4) за кожною лабораторною роботою складається звіт.

Методичні вказівки підготовлені на кафедрі «Обчислювальна техніка та програмування» і можуть бути використані для підготовки дипломованих фахівців за спеціальністю «Комп'ютерна інженерія».

Лабораторна робота 1
ОЗНАЙОМЛЕННЯ З ПАКЕТОМ MPLAB ТА СКЛАДАННЯ ПРОГРАМ МОВОЮ АСЕМБЛЕРА

Мета роботи: ознайомитися з інтегрованим середовищем для розробки та налагодження програм MPLAB, вивчити структуру програми, навчитися виконувати налагодження програми, набути практичних навичок роботи з пакетом MPLAB.

Індивідуальне завдання

На прикладі мікроконтролера PIC16C56 написати програму для запису числа *M* у комірку пам'яті за адресою *A* згідно з варіантом (табл. 1.1).

Таблиця 1.1

Номер варіанта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	23	24	25	26	27	28	29	30	31	9	10	11	12	13	14	15	16	17
M	37	65	93	86	180	34	48	201	74	80	207	176	18	55	156	238	49	50

Структура програми:

Секція заголовка:

– назва програми;

– прізвище та ім'я автора програми;

– дата створення програми;

– опис ресурсів мікроконтролера (МК), а також змінних і констант, що використовуються в програмі (їх особливості);

– підключення файлу з описом регістрів спеціального призначення.

Робоча секція:

– ініціалізація МК;

– основна програма;

Секція закінчення

Приклад тексту програми:

```
; Секція заголовка
; заголовок програми
; Лабораторна робота ___ (вказати номер і тему роботи)
; Виконав студент групи КІТ–___ (вказати групу, прізвище та ініціали)
; Варіант __ (вказати номер варіанта)
; Завдання (вказати коротко завдання та вихідні дані)
; Дата ___ Версія ___ (вказати дату виконання роботи і номер версії)
; підключення файлу з описом регістрів спец. призначення
#include <p16c56.inc>          ; для версії 8.66 і більш старших
#include <p16c5x.inc>          ; для більш ранніх версій
; опис змінних
A  equ   .11          ; значення з таблиці завдань
M equ   .87           ; значення з таблиці завдань
; Робоча секція
         ORG  0       ; точка входу – вектор скидання
; ініціалізація МК
; основна програма
         movlw    M          ; записати константу в акумулятор
         movwf    A          ; запис із акумулятора в пам’ять даних
loop     goto     loop       ; останов програми
; Секція закінчення
         END
```

Налагодження програми:

– вибір кристала PIC16C56 (меню Configure, Select Device);

– вибір режиму симулятора в меню Debugger (MPLAB – SIM);

– компіляція програми (меню Project – Quickbuild);

– усунення помилок і попереджень;

– покрокове виконання програми (Debugger – Step);

– перегляд пам’яті програм (меню View – Program Memory);

– перегляд спеціальних регістрів (Special Function Registers);

– перегляд пам’яті даних (File Registers).

Зміст звіту

1. Тема.
1. Мета.
2. Індивідуальне завдання.
3. Алгоритм програми з необхідними коментарями.
4. Лістинг програми (файл *.lst).
5. Методика тестування програми та зміст *.sti файлу.
6. Результати виконання програми (вікна пакета MPLAB з необхідними поясненнями; в них обов’язково має бути присутнім фрагмент програми із заголовком).
7. Висновки.

Лабораторна робота 2
ОРГАНІЗАЦІЯ ПАРАЛЕЛЬНОГО ІНТЕРФЕЙСУ

Мета роботи: вивчити організацію паралельного інтерфейсу та виконати налагодження програми для уведення інформації від датчиків дискретних сигналів.

Індивідуальне завдання

Число *M* (див. табл. 1.1 у лабораторній роботі 1) за допомогою «асинхронних стимулів» подати в порт *B* мікроконтролера. Написати програму, що зчитує код з порту *B*, який потім виводить у порт *A* по тетрадах.

Наприклад, число $M = 87_{10} = 01010111_2$. У меню Debugger – Stimulus – New Workbook у колонці Pin/SFR по черзі вибираємо виводи порту *B*, що відповідають одиницям у числі *M* (RB0, RB1, RB2, RB4, RB6), потім у колонці Action вибираємо Set High. Після того, як всі виводи задані, необхідно застосувати зміни (кнопка Apply) і зберегти у файлі. Потім натиснути на кнопці «>» у колонці Fire для кожного заданого рядка (рис. 2.1).

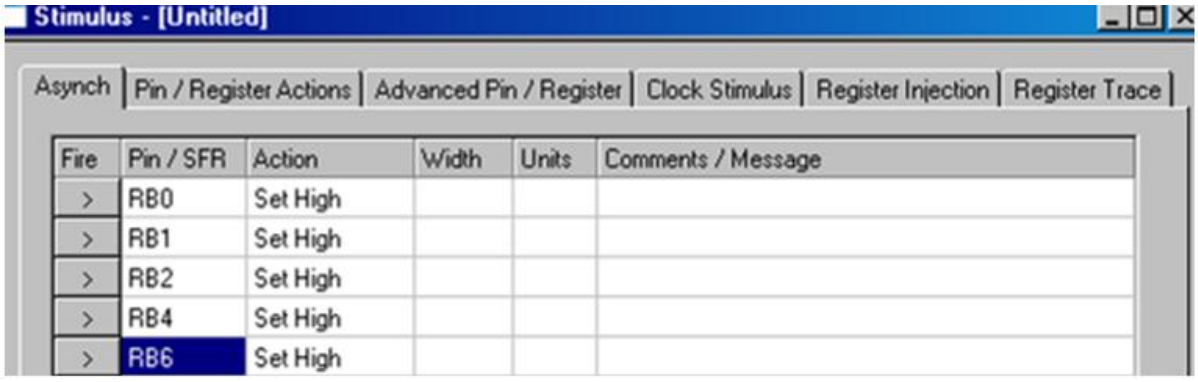


Рисунок 2.1

Приклад тексту програми:

; Секція заголовка

; заголовок програми

```
; підключення файлу з описом регістрів спец. призначення
#include <p16c56.inc>          ; для версії 8.66
#include <p16c5x.inc>          ; для більш ранніх версій
; опис змінних і констант
A       equ   .11    ; адреса комірки пам'яті
                     ; значення взяти з табл. 1.1 в лабораторній роботі 1
; Робоча секція
; текст основної програми
        ORG   0      ; точка входу – вектор скидання
; ініціалізація МК
        movlw   0
        tris    PORTA     ; налагодження порту А на виведення
        movlw   0xFF
        tris    PORTB     ; налагодження порту В на уведення
; основна програма
        movf    PORTB, 0  ; читання порту В у акумулятор
        movwf   A         ; запис із акумулятора у пам'ять даних
        movwf   PORTA     ; виведення у порт А молодшої тетради,
; старші розряди в порту А не існують
        swapf   A, 0      ; обмін тетрад у комірці пам'яті А, результат
                          ; буде поміщений в акумулятор (W)
        movwf   PORTA     ; виведення у порт А старшої тетради
                          ; введеного числа
loop    goto    loop      ; останов програми
; Секція закінчення
        END
```

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Лабораторна робота 3
РЕЖИМИ РОБОТИ ТАЙМЕРА (TMR0)

Мета роботи: вивчити основні режими функціонування таймера, способи та особливості його ініціалізації, варіанти використання та налагодження переддільника.

Індивідуальне завдання

Написати програму, в якій реалізувати розподіл зовнішньої частоти F_1 з коефіцієнтом розподілу $K=(C+N)\cdot 10$, де C – номер групи; N – номер варіанта. Сигнал з частотою F_1 надходить на вивід RA4/T0CKI мікроконтролера. Для парних варіантів – рахунковий перепад по фронту імпульсу, для непарних – по спаду імпульсу. Сигнал із частотою $F_2 = F_1 / K$ вивести на вивід RB_i, де $i = N \cdot \mathrm{mod}8$. Оцінити максимально можливу вхідну частоту F_1.

Виконання ініціалізації таймера TMR0:

– визначення значення бітів T0CS, T0SE, PSA, (біти 5 – 3 у регістрі OPTION);

– визначення значення бітів PS2 – PS0 (біти 2 – 0 у регістрі OPTION).

Приклад фрагмента програми, що виконує конфігурацію схеми таймера за такими вихідними даними: тактування таймера здійснюється від зовнішнього джерела тактових імпульсів по задньому фронту з використанням переддільника з коефіцієнтом розподілу 1:64:

```
…
   movlw     b’00110101’
             ; ||||||||
             ; |||||└└└——— коефіцієнт переддільника 1:64
             ; ||||└———— переддільник підключений до каналу таймера
             ; |||└———— тактування по задньому фронту
             ; ||└———— тактування від зовнішнього генератора
             ; └└———— у даній роботі не мають значення
   option    ; конфігурація схеми таймера
   …         ; продовження програми
```

Виконання ділення зовнішньої частоти F_1

Приклад формування на виході початкового рівня сигналу:

…

bcf PORTB, 5 ; установка початкового рівня вихідного сигналу

…

Виконання затримки на K / 2 періодів частоти F_1 за допомогою таймера

Приклад фрагмента програми, що виконує підрахунок 10 періодів з виводу RA4/T0CKI за допомогою таймера:

```
    …
    bcf    INTCON, T0IF  ; скидання прапорця переповнення
                         ; таймера
    movlw  .256-10       ; формування константи для
    movwf  TMR0          ; ініціалізації таймера
m1  btfss  INTCON, T0IF  ; очікування переповнення таймера
    goto   m1            ; перехід, якщо таймер не переповнений
    …                    ; продовження програми
```

Приклад зміни на виході рівня сигналу на протилежний:

…

bsf PORTB, 5 ; зміна рівня вихідного сигналу

…

Приклад повторення дій, починаючи з формування початкового рівня:

…

goto loop ; перехід на мітку loop, яка повинна вказувати на команду bcf

…

Порядок налагодження програми

Задання вхідної частоти F_1

На вивід RA4/T0CKI мікроконтролера необхідно подати вхідну частоту. В меню Debugger – Stimulus – New Workbook на вкладці Clock Stimulus у колонці Pin вибираємо вивід T0CKI, початкове значення (Initial) – Low, тривалість нульового рівня (Low Cycles) – 10, тривалість одиничного рівня (High Cycles) –

10, почати (Begin) – At Start, закінчити (End) – Never. Таким чином, на вивід RA4/T0CKI буде надходити сигнал із періодом $T_1 = 10 + 10 = 20$ (тактів частоти синхронізації мікроконтролера), а на виводі RB_i має формуватися сигнал із періодом $T_2 = K \cdot T_1$.

Перевірка роботи схеми таймера

У віконці SFR контролюємо зміну регістрів PORTA, TMR0 і INTCON. У покроковому режимі після виконання кожних 10 командних тактів (команда goto виконується за два такти, інші – за один такт) повинне змінювтися значення четвертого розряду порту А (RA4). Після кожної зміни $0 \rightarrow 1$ або $1 \rightarrow 0$ четвертого розряду порту *A* залежно від налагодження таймера (рахунковий перепад «фронт» ($0 \rightarrow 1$) або «спад» ($1 \rightarrow 0$) – біт T0SE у регістрі OPTION) має збільшуватися значення регістра TMR0. При переповненні таймера ($255 \rightarrow 0$) установлюється прапор переповнення – другий розряд (T0IF) у регістрі INTCON.

Використання «Секундоміра» (StopWatch)

У меню Debugger вибираємо Stopwatch. На команді bcf або bsf ставимо «точку останова» (Breakpoint). Запускаємо програму (Debugger – Run), чекаємо останова. Якщо допущено помилку при налагодженні таймера або формуванні сигналу на виводі RA4/T0CKI (Clock Stimulus), програма буде виконуватися безупинно. Після останова обнуляємо секундомір (кнопка Zero в Stopwatch) та повторюємо запуск програми. Після останова на «точці останова» контролюємо значення Instruction cycles в Stopwatch. Воно має дорівнювати $T2 \pm 3$ такти. Обнуляємо секундомір і повторюємо запуск.

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Лабораторна робота 4
СТОРОЖОВИЙ ТАЙМЕР (WDT)

Мета роботи: вивчити функціонування сторожового таймера (WDT), способи та особливості його налагодження та використання, підключення переддільника.

Індивідуальне завдання

Дослідити функціонування сторожового таймера в режимі SLEEP; визначити період скидання сторожового таймера при використанні переддільника з коефіцієнтом розподілу $K = 2^n$, при $n = N \bmod 8$, де N – номер варіанта.

Порядок налагодження програми

1. Виконати ініціалізацію портів уведення/виведення.

2. Виконати ініціалізацію сторожового таймера WDT:

дозволити функціонування сторожового таймера в конфігураційному слові мікроконтролера, для цього в меню Configure середовища MPLAB у розділі Configuration Bits значення поля WDTE встановити WDT enabled; визначити значення біта PSA (біт 3 у регістрі OPTION); визначити значення бітів PS2-PS0 (біти 2 – 0 у регістрі OPTION).

Приклад фрагмента програми, що виконує конфігурацію схеми сторожового таймера за такими вихідними даними (використовується переддільник із коефіцієнтом розподілу 1:32):

```
…
movlw     b’00001101’
          ; ||||||||
          ;|| ||| └└└──── коефіцієнт переддільника 1:32
          ;|| ||└──────── переддільник підключений до сторожового таймера
          ;└└└└────────── у даній роботі не мають значення
option    ; конфігурація схеми таймера
…         ; продовження програми
```

3. Виконати перевірку функціонування сторожового таймера:

– організувати в програмі нескінченний цикл, наприклад:

```
…
 nop          ; пуста команда
 goto $       ; перехід на адресу поточної команди
 nop          ; пуста команда
…
```

– встановити «точку останова» (Breakpoint) на команди nop;

– запустити програму для виконання (Debugger – Run);

– упевнитися, що виконання програми завжди зупиняється на першій команді nop та ніколи – на другій;

– за допомогою «Секундоміра» (StopWatch) виміряти період скидання сторожового таймера;

– добавити у програму команду скидання сторожового таймера – clrwdt, наприклад:

```
      …
      nop          ; пуста команда
loop clrwdt        ; скидання сторожового таймера
      goto loop    ; перехід на мітку loop
      nop          ; пуста команда
      …
```

– переконатися, що виконання програми не переривається за сторожовим таймером.

4. Виконати перевірку функціонування сторожового таймера в режимі Sleep:

– замість нескінченного циклу виконати перехід у режим зниженого енергоспоживання «сон», наприклад:

```
…
```

```
nop         ; пуста команда
sleep       ; перехід у режим «сна»
nop         ; пуста команда
…
```

– встановити «точки останова» (Breakpoint) на команді nop;

– у покроковому режимі перевірити роботу програми та період скидання сторожового таймера.

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт.

Лабораторна робота 5
ОРГАНІЗАЦІЯ ПАМ'ЯТІ ДАНИХ І ПРОГРАМ

Мета роботи: вивчити сторінкову організацію і способи адресації пам'яті програм. Навчитися здійснювати явний та неявний виклик підпрограм.

Індивідуальне завдання

Написати програму мовою асемблера для мікроконтролера PIC16F84, в якій зробити таке:

– виконати ініціалізацію мікроконтролера (зробити необхідні налагодження елементів, вузлів і модулів мікроконтролера, що використовуються для розв'язання поставленої задачі);

– розмістити в нульовій сторінці пам'яті програм за адресою $A_i = N + 100$ (N – номер варіанта) підпрограму, що виконує запис константи M у комірку пам'яті даних A (див. лабораторну роботу 1), виклик підпрограми здійснювати явно;

– розмістити в першій (якщо вона є) або в нульовій (якщо першої немає) сторінці пам'яті програм за адресою $A_i = N + 200$ підпрограму для організації паралельного інтерфейсу, що виконує читання порту B і запис його вмісту в пам'ять даних та виведення цих даних у порт A (див. лабораторну роботу 2), виклик підпрограми здійснювати неявно;

– розмістити в нульовій сторінці пам'яті програм за адресою $A_i = N + 300$ підпрограму, що виконує за допомогою таймера ділення зовнішньої частоти F на коефіцієнт K_1 і виводить отриману частоту F / K_1 на вивід RB0 мікроконтролера (див. лабораторну роботу 3), виклик підпрограми здійснювати явно.

Індивідуальні варіанти завдань, окрім початкових адрес підпрограм, наведено в лабораторних роботах 1 – 3.

Приклад фрагмента програми, що виконує явний виклик підпрограми:

```
…              ; основна програма
call    proc1  ; виклик підпрограми proc1
```

```
…               ; продовження основної програми
proc1           ; підпрограма proc1
…               ; команди підпрограми
return          ; повернення з підпрограми в основну програму
```

Приклад фрагмента програми, що виконує неявний виклик підпрограми адресою 0x345:

```
…               ; основна програма
call    proc2   ; виклик підпрограми proc2
…               ; продовження основної програми
proc2           ; підпрограма proc2
movlw   0x03    ; запис старших розрядів адреси 0x345
movwf   PCLATH  ; у регістр PCLATH
movlw   0x45    ; запис молодших розрядів адреси 0x345
movwf   PCL     ; у регістр PCL і перехід за адресою 0x345
…
org     0x345   ; розміщення підпрограми за адресою 0x345
…               ; команди підпрограми
return          ; повернення з підпрограми в основну програму
```

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Лабораторна робота 6
ОРГАНІЗАЦІЯ І ВИКОРИСТАННЯ ПАМ'ЯТІ ДАНИХ EEPROM

Мета роботи: вивчити організацію і способи доступу до енергонезалежної пам'яті даних (EEPROM).

Індивідуальне завдання

Написати програму мовою асемблера для мікроконтролера PIC16F84, в якій зробити таке:

– виконати ініціалізацію мікроконтролера (зробити необхідні налаштування елементів, вузлів та модулів мікроконтролера, що використовуються для розв'язання поставленої задачі) без використання команд TRIS і OPTION, а використовуючи спеціальні регістри банку 1;

– за допомогою методу прямої адресації записати в пам'ять даних мікроконтролера своє прізвище, ім'я та по батькові;

– за допомогою методу непрямої адресації переписати прізвище, ім'я та по батькові з пам'яті даних в енергонезалежну пам'ять (EEPROM);

– в останню комірку енергонезалежної пам'яті даних записати номер варіанта;

– визначити час запису одного байта в EEPROM;

– визначити можливість читання даних з EEPROM відразу після початку циклу запису.

Приклад фрагмента програми, що виконує очищення 10 байтів пам'яті даних, починаючи з адреси 0x0C, використовуючи метод непрямої адресації:

```
…
movlw  0x0C     ; встановлення початкової адреси
movwf  FSR      ; ініціалізація вказівника
movlw  .10      ; ініціалізація лічильника CTR значенням 10
movwf  CTR
```

```
m1  clrf    INDF    ; очищення комірки пам’яті даних
    incf    FSR     ; збільшення значення вказівника
    decfsz  CTR, 1  ; зменшення значення лічильника
    goto    m1      ; перехід, якщо не остання комірка
    …               ; продовження програми
```

Зауваження. Лічильник CTR не повинен розташовуватися в комірках пам’яті, що очищуються. Імена регістрів FSR (адреса 0x00) і INDF (адреса 0x04) доступні при підключенні заголовного файлу P16F84.INC або повинні бути описані в описовій секції програми.

Приклад програми, що виконує зчитування даних з комірки пам’яті EEPROM з адресою 0x10 у регістр W:

```
    …
    bcf     STATUS, RP0     ; вибір банку 0
    movlw   0x10            ; визначення адреси комірки пам’яті EEPROM
    movwf   EEADR
    bsf     STATUS, RP0     ; вибір банку 1
    bsf     EECON1, RD      ; строб читання
    bcf     STATUS, RP0     ; вибір банку 0
    movf    EEDATA, W       ; запис у регістр W результату читання
    …
```

Приклад програми, що виконує запис значення регістру W у комірку пам’яті EEPROM з адресою 0x10:

```
    …
    bcf     STATUS, RP0     ; вибір банку 0
    movwf   EEDATA          ; дані для запису
    movlw   0x10            ; визначення адреси комірки пам’яті EEPROM
    movwf   EEADR
    bsf     STATUS, RP0     ; вибір банку 1
```

```
    bsf     EECON1, WREN ; дозвіл запису
    bcf     EECON1, EEIF  ;– скидання прапорця закінчення запису в EEPROM
    bcf     INTCON, GIE   ;+ заборона переривань
    movlw   0x55          ;+ обов’язкова послідовність команд
    movwf   EECON2        ;+
    movlw   0xAA          ;+
    movwf   EECON2        ;+
    bsf     EECON1, WR    ;+ строб запису
m1  btfss   EECON1, EEIF  ;– очікування закінчення запису в EEPROM
    goto    m1            ;–
    bcf     STATUS, RP0   ; вибір банку 0
    …
```

Зауваження. Послідовність команд, що відзначені символом «+» є обов’язковою. Команди, що відзначені символом «–» є необов’язковими у випадку однократного запису. Імена регістрів STATUS (адреса 0x03), EEDATA (адреса 0x08), EEADR (адреса 0x09), INTCON (адреса 0x0B), EECON1 (адреса 0x88), EECON2 (адреса 0x89), і біт RP0(5), RD(0), WR(1), WREN(2), EEIF(4), GIE(7) доступні при підключенні заголовного файлу P16F84.INC або повинні бути описані в описовій секції програми.

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Лабораторна робота 7
СИСТЕМА ПЕРЕРИВАНЬ МІКРОКОНТРОЛЕРА PIC16F84. ВЛАСНІ ОБРОБЛЮВАЧІ ПЕРЕРИВАНЬ

Мета роботи: вивчити систему переривань мікроконтролера PIC16F84, способи формування переривань, використання оброблювачів декількох переривань.

Індивідуальне завдання

Написати програму для мікроконтролера PIC16F84, в якій зробити таке:

1) виконати необхідні налагодження елементів, вузлів та модулів мікроконтролера;

2) написати оброблювачі переривань:

– оброблювач переривання після зміни рівня сигналу на виводі RB0/INT має виконувати запис константи у комірку пам’яті даних та виведення цих даних у порт *B*, парні варіанти обробляють передній фронт сигналу на виводі RB0/INT, непарні – задній фронт;

– оброблювач переривання після зміни рівня сигналу на виводах RB4-RB7 має виконувати читання порту *B* та виведення цих даних у порт *A*;

– оброблювач переривання після переповнення таймера TMR0 має виконувати розподіл частоти за фіксованим коефіцієнтом;

– оброблювач переривання після закінчення запису даних в енергонезалежну пам’ять (EEPROM) має виконувати запис константи у наступну комірку пам’яті EEPROM, значення константи визначається як номер варіанта.

Приклад фрагмента програми, що виконує збереження регістрів при виклику переривання, а також виклик оброблювача переривання після зміни рівня сигналу на виводах RB4-RB7:

```
W_TEMP      EQU   0x0C          ; адреса регістру збереження W
STATUS_TEMP EQU   0X0D          ; адреса регістру збереження STATUS
A           EQU   0x0E          ; адреса регістру збереження стану порту B
  int_point  org   0x04         ; вектор переривань
   movwf     W_TEMP             ; збереження W
   swapf     STATUS,W           ; збереження STATUS
   movwf     STATUS_TEMP
   btfsc     INTCON,RBIF        ; переривання при зміні RB4-RB7
   call      INTRB47            ; виклик оброблювача переривання
                                ; при зміні сигналу на виводах RB4-RB7
   swapf     STATUS_TEMP,W ; відновлення регістру STATUS
   movwf     STATUS
   swapf     W_TEMP, F          ; відновлення регістру W
   swapf     W_TEMP, W
   retfie                       ; повернення із переривання
   INTRB47                      ; оброблювач переривання при зміні сигналу
                                ; на виводах RB4-RB7
   movf      PORTB, W           ; читання стану порту B
   movwf     PORTA              ; виведення молодшої тетради у PORTA
   movwf     A                  ; збереження у пам’яті
   swapf     A, W               ; виведення старшої тетради у PORTA
   bcf       INTCON,RBIF        ; очищення прапорця переривання
   return
   …                            ; продовження програми
```

Зауваження. Імена регістрів спеціального призначення доступні при підключенні заголовного файлу P16F84.INC, або повинні бути описані в секції заголовка програми.

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Лабораторна робота 8
ВВЕДЕННЯ АНАЛОГОВИХ СИГНАЛІВ У МІКРОКОНТРОЛЕРИ PIC16C71x / PIC16F8xx

Мета роботи: вивчити принцип роботи аналого-цифрового перетворювача на прикладі мікроконтролера PIC16C71. Навчитися вводити аналогові сигнали.

Індивідуальне завдання

1. Виконати конфігурацію мікроконтролера для введення аналогових сигналів.
2. Ввести сигнали з двох аналогових датчиків U1 і U2.
3. Зробити порівняння введених сигналів.
4. За результатами порівняння виконати підпрограму згідно з номером варіанта (табл. 8.1).
5. Визначити час виконання перетворення АЦП.

Таблиця 8.1

Номер варіанта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
U1 < U2	A	B	C	D	E	F	G	H	I	J	A	B	C	D	E	F	D	C	I	J
U1 = U2	I	J	G	H	E	F	D	C	B	J	A	I	C	G	F	E	G	H	I	A
U1 > U2	A	J	C	H	B	D	G	C	I	E	J	B	H	D	E	E	D	H	B	J

Дії підпрограм:

A – інкремент комірки пам’яті даних;

B – декремент 16-розрядного лічильника в пам’яті даних;

C – зчитування 8-розрядного числа з порту *B* та запис його в комірку пам’яті даних;

D – запис поточного значення таймера (регістр TMR0) в комірку пам’яті даних;

E – формування позитивного імпульсу тривалістю $t_і = 5 \cdot t_ц$ на виводі RB0, де $t_ц$ – тривалість командного циклу;

F – інкремент 16-розрядного лічильника в пам’яті даних;

G – одночасне інвертування сигналів на виводах RB0 і RB1;

H – декремент комірки пам’яті даних;

I – обмін рівнів сигналів на виводах RB1 та RB2;

J – прийом дискретного сигналу з виводу RB3 і видача його на вивід RB0.

Тактування мікроконтролера виконати від тактового генератора з частотою f_{osc} = 4 МГц.

Стислі теоретичні відомості

Ввід аналогових сигналів виконується за допомогою аналого-цифрового перетворювача (АЦП), для керування яким у мікроконтролері PIC16C71 необхідно використовувати такі регістри:

– регістр результату (ADRES);

– регістр керування 0 (ADCON0);

– регістр керування 1 (ADCON1);

– регістр керування перериваннями (INTCON).

У регістрі ADRES зберігається результат аналого-цифрового перетворення. Коли перетворення завершене, результат перетворення записується в регістр ADRES, після цього скидається прапорець GO/DONE (ADCON0<2>) і встановлюється прапорець переривання ADIF (ADCON0<1>).

Потім необхідно витримати паузу 20 – 30 мкс для розряду ємності схеми вибірки-збереження АЦП (це необхідно повторювати після кожного наступного циклу перетворення). Тільки після цього потрібно встановити біт GO/DONE (ADCON0<2>) для початку перетворення.

Алгоритм дій для виконання аналого-цифрового перетворення:

1. Здійснити конфігурацію (ініціалізацію) модуля АЦП:

– вибрати аналогові входи та джерело опорної напруги (біти PCFG2 – PCFG0 у регістрі ADCON1);

– вибрати вхідний канал АЦП (біти CHS2 – CHS0 у регістрі ADCON0);

– вибрати джерело та частоту імпульсів перетворення (біти ADCS1, ADCS0 у

регістрі ADCON0);

– увімкнути модуль АЦП (біт ADON у регістрі ADCON0).

2. Налагодити переривання від модуля АЦП (за необхідності):

– скинути прапорець закінчення переривання (біт ADIF у регістрі ADCON0);

– дозволити переривання від АЦП (біт ADIE у регістрі INTCON);

– дозволити всі переривання (біт GIE у регістрі INTCON).

3. Витримати паузу 20 – 30 мкс.

4. Почати аналого-цифрове перетворення:

– встановити біт запуску АЦП (біт GO/DONE у регістрі ADCON0).

5. Очікувати закінчення перетворення, наприкінці якого автоматично виконується:

– скидання біта закінчення перетворення (біт GO/DONE у регістрі ADCON0);

– формування сигналу (установка біта ADIF у регістрі ADCON0).

6. Отримати результати перетворення:

– прочитати результат перетворення (регістр ADRES);

– скинути, якщо необхідно, прапорець закінчення переривання (біт ADIF у регістрі ADCON0).

7. Виконати дії, починаючи з п. 1 або п. 2, для виконання наступного перетворення.

Приклад фрагмента програми, що виконує налагодження АЦП і аналого-цифрове перетворення вхідного сигналу з виводу RA2/AN2.

```
; підключення файлу з описом стандартних констант і значень
#include p16c71.inc
…
; налагодження АЦП
initAD
        bsf             STATUS, RP0             ; вибір банку 1
        movlw           b’00000000’             ; RA3-RA0 аналогові входи
```

```
        movwf   ADCON1
        bcf     STATUS, RP0         ; вибір банку 0
        movlw   b'11010001'         ; RC-генератор для АЦП,
        movwf   ADCON0              ; канал 2, дозвіл АЦП переривання
Convert
        call    Delay30us           ; затримка 30 мкс
        bsf     ADCON0, GO_DONE     ; запуск АЦП
loop
        btfsc   ADCON0, GO_DONE     ; перетворення закінчене?
        goto    loop                ; продовжити чекання
        movf    ADRES, W            ; результат перетворення в W
        …                           ; продовження програми
```

Моделювання роботи АЦП в пакеті MPLAB

1. Створити текстовий файл із передбачуваними значеннями результатів перетворення АЦП, наприклад adcres.reg (рис. 8.1).

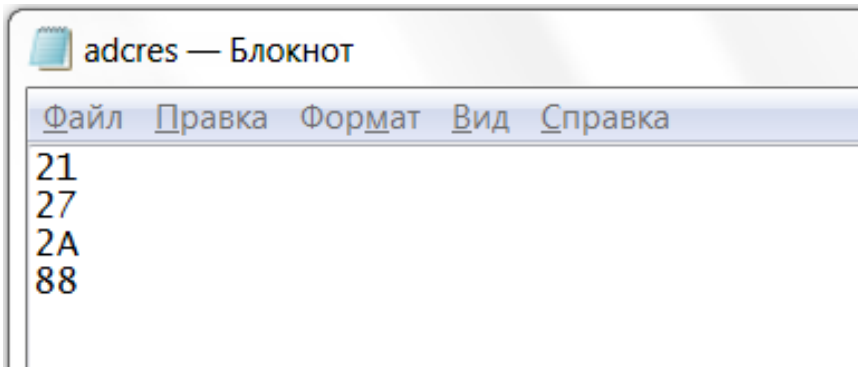


Рисунок 8.1 – Створення текстового файлу з результатами перетворення АЦП

2. Відкрити вікно Register Stimulus через меню: Debug–>Simulator Stimulus–> Register Stimulus–>Enable...

3. У полі Program Memory Address вказати адресу команди в пам'яті програм (дозволяється і рекомендується вказувати мітку), наприклад, ares.

4. У полі Register Address вкажіть адресу або символьне ім’я регістру, наприклад, ADRES.

5. Натисніть кнопку Browse для зазначення файлу (наприклад, adcres.reg).

6. Після скидання мікроконтролера, коли значення лічильника команд PC збігається з указаною адресою ares, у регістр ADRES буде занесене значення з файлу adcres.reg. При досягненні кінця файлу adcres.reg підстановка почнеться з початку файлу, тобто буде виконуватися циклічно.

Після виконання програми необхідно виконати збереження всіх отриманих результатів та скласти звіт про виконану лабораторну роботу.

Список літератури

1. Скороделов В. В. Организация и программирование микроконтроллеров : учебник / В. В. Скороделов, И. А. Фурман, В. А. Краснобаев, А. Н. Рысованый. – Харьков : Эспада, 2006. – 248 с.

2. Скородєлов В. В. Цифрові пристрої та мікропроцесори. Архітектура та програмування мікроконтролерів: навчальний посібник / В. В. Скородєлов, О. М. Рисований, О. Ф. Даниленко, М. В. Ліпчанський. – Харків : ХВУ, 2004. – 318 с.

3. Катцен С. PIC-микроконтроллеры. Все, что вам необходимо знать / С. Катцен. – Москва : Додека-XXI, 2008. – 656 с.

4. Брей Б. Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением С и ассемблера / Б. Брей. – Киев : МК-Пресс, 2008. – 576 с.

5. Уилмсхерст Т. Разработка встроенных систем с помощью микроконтроллеров PIC. Принципы и практические примеры / Т. Уилмсхерст. – Киев: МК-Пресс, 2008. – 544 с.

6. Болл С. Р. Аналоговые интерфейсы микроконтроллеров / С. Р. Болл. – Москва : Додека–XXI, 2008. – 656 с.

7. Яценков В. С. Микроконтроллеры Microchip с аппаратной поддержкой USB / В. С. Яценков. – Москва : Горячая линия. – Телеком, 2008. – 400 с.

8. Скороделов В. В. Проектирование устройств на однокристальных микроконтроллерах с RISC-архитектурой. Книга 1. Особенности проектирования и архитектура микроконтроллеров PIC: учебное пособие. – Харьков : ХГПУ, 1999. – 120 с.

9. Скороделов В. В. Проектирование устройств на однокристальных микроконтроллерах с RISC-архитектурой. Книга 2. Разработка и отладка программ для ОМК PIC : учебное пособие. – Харьков : ХГПУ, 1999. – 127 с.

ЗМІСТ

Навчальне видання

Методичні вказівки до виконання лабораторних робіт
з навчальної дисципліни «Вбудовані системи»
для студентів денної та заочної форм навчання
за спеціальністю «Комп’ютерна інженерія»

Укладачі:
ЛІПЧАНСЬКИЙ Максим Валентинович,
СКОРОДЄЛОВ Володимир Васильович,
ГЕЙКО Геннадій Вікторович

Відповідальний за випуск проф. С. Г. Семенов
Роботу до видання рекомендував проф. М. Й. Заполовський
Редактор О. В. Козюк

План 2019 р., поз. 79
Підп. до друку 24.06.2020 р. Формат 60х84 1/16.
Папір офсет. Друк ризографічний. Ум. друк. арк. 1,5.
Наклад 50 прим. Замовлення № 625-20

Видавець:
Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво суб’єкта видавничої справи ДК № 5478 від 21.08.2017 р.
